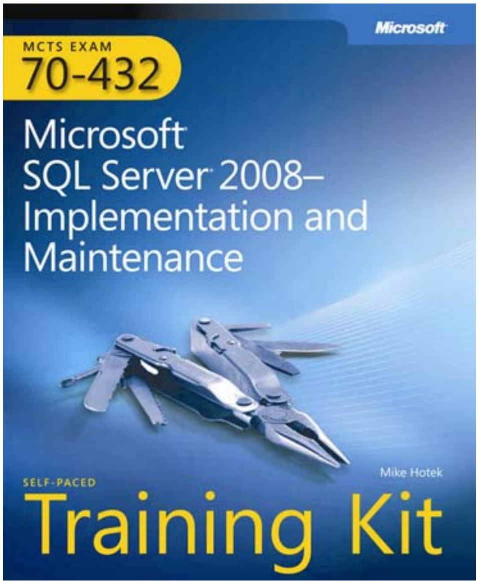


MCTS Self-Paced Training Kit (Exam 70-432): Microsoft® SQL Server® 2008—Implementation and Maintenance

Mike Hotek

To learn more about this book, visit Microsoft Learning at
http://www.microsoft.com/MSPress/books/12858.aspx

9780735626058

Contents

Chapter 8 Designing Policy Based Management 177

CHAPTER 8

Designing Policy Based Management

Prior to Microsoft SQL Server 2008, you performed configuration management of an environment by using a conglomeration of documents, scripts, and manual checking. The configuration options, naming conventions, and allowed feature set were outlined in one or more documents. To enforce your standards, you would have had to connect to each instance and execute scripts that needed to be maintained and updated with new versions and service packs. In this chapter, you learn about the new Policy Based Management framework that allows you to check and enforce policy compliance across your entire SQL Server infrastructure.

Exam objectives in this chapter:

- Implement the declarative management framework (DMF).
- Configure surface area.

Lesson in this chapter:

Before You Begin

To complete the lessons in this chapter, you must have:

- SQL Server 2008 installed
- The *AdventureWorks* database installed within the instance

REAL WORLD

Michael Hotek

Managing a single server running SQL Server or even a small group of them, one at a time, has always been reasonably straightforward. However, when you needed to uniformly manage an entire SQL Server environment or a large group of instances, you had to either write a large amount of custom code or purchase additional products.

One customer I work with has an environment with more than 5,000 SQL Server instances. Prior to the release of SQL Server 2008, two DBAs were required to manage the almost 50,000 lines of code that checked instances for compliance to corporate policies. They devoted more than 70 hours each week to maintaining the code and checking systems.

After deploying SQL Server 2008, they started to convert all their code to policies. After the conversion was completed, they estimate that less than 1,000 lines of custom logic remained. By using the central management features to check and enforce policies across the environment, they should be able to save over 3,000 hours of management and maintenance time per year.

Lesson 1: Designing Policies

SQL Server 2008 has a new feature called Policy Based Management, also known as the declarative management framework (DMF), to tackle the problem of standardizing your SQL Server instances. Although Policy Based Management can be used just to alert an administrator when an object is out of compliance, depending upon the type of policy, you can also enforce compliance by preventing changes that would violate a policy.

Policy Based Management introduces the following new objects that are used to design and check for compliance:

- Facets
- Conditions
- Policies
- Policy targets
- Policy categories

After this lesson, you will be able to:

- Create conditions
- Define policies
- Specify targets for policy checking
- Configure policy categories
- Check for policy compliance
- Import and export policies

Estimated lesson time: 30 minutes

Facets

Facets are the core object upon which your standards are built. Facets define the type of object or option to be checked, such as database, Surface Area, and login. SQL Server ships with 74 facets, implemented as .NET assemblies, each with a unique set of properties.

All the objects for Policy Based Management are stored within the *msdb* database. You can get a list of the facets available by querying the dbo.syspolicy_management_facets table. Unfortunately, unless you want to write code to interact with Server Management Objects (SMOs), the only way to get a list of facet properties is to open each facet in SQL Server Management Studio (SSMS), one at a time, and view the list of properties.

Conditions

When you define a WHERE clause for a data manipulation language (DML) statement, you set a condition for the DML statement that defines the set of rows that meet your specific inclusion criteria. Within the Policy Based Management framework, *conditions* are the equivalent of a *WHERE* clause that defines the criteria needing to be checked.

You define the conditions that you want to check or enforce for a policy by defining criteria for the properties of a facet. Just like a WHERE clause, a condition can be defined by one or more facet properties, and a single facet property can be checked for multiple criteria. The comparison operators that can be used are restricted by the data type of the property. For example, a property of type *string* can be checked with =, <>, *LIKE, NOT LIKE, IN,* or *NOT IN,* whereas a boolean type can only be checked for = and <>.

If a condition that you want to check for a facet does not have a specific property that can be used, you can use the advanced editor to define complex conditions that compare multiple properties and incorporate functions. For example, you can check that every table has a primary key and that a table with a single index must be clustered. Unfortunately, if you define a condition using the advanced editor, a policy that incorporates the condition must be executed manually and cannot be scheduled.

Conditions are checked in a single step. You cannot have a condition pull a list of objects, iterate across the list of objects, and then apply subsequent checks. To work within the Policy-Based Management framework, conditions need to return a True or False value. Therefore, when building complex conditions with the advanced editor, you cannot return a list of objects that do not meet your criteria. You have to define the condition such that if any object does not meet your criteria, a value of False is returned.

Although you can check many properties of a facet within a single condition, a single condition can't be defined for multiple facets. For example, you can check all 10 of the properties for the Surface Area Configuration facet in a single condition, but you have to define a second condition to check a property of the Surface Area Configuration for Analysis Services.

Policy Targets

Conditions are the foundation for policies. However, you don't always want to check policies across every object available, such as every database in an instance or every index within every database. Conditions can also be used to specify the objects to compare the condition against, called *policy targeting* or target sets.

You can target a policy at the server level, such as instances that are SQL Server 2005 or SQL Server 2008. You can also target a policy at the database level, such as all user databases or all system databases.

Policies

Policies are created for a single condition and set to either enforce or check compliance. The execution mode can be set as follows:

- **On demand** Evaluates the policy when directly executed by a user
- **On change, prevent** Creates data definition language (DDL) triggers to prevent a change that violates the policy
- **On change, log only** Checks the policy automatically when a change is made using the event notification infrastructure
- **On schedule** Creates a SQL Server Agent job to check the policy on a defined schedule

If a policy contains a condition that was defined using the advanced editor, the only available execution mode is On Demand.

To use the *On change, prevent* and *On change, log only* execution modes, the policy must target instances running SQL Server 2005 and above. The *On change, log only* execution mode uses the event notification infrastructure that is available only for SQL Server 2005 and later. The *On change, prevent* execution mode depends on DDL triggers to prevent a change that is not in compliance with the policy and are available only for SQL Server 2005 and later. In addition, you can set a policy to *On change, prevent* only if it is possible for a DDL trigger to prevent the change. For example, you could prevent the creation of an object that violated your naming conventions, but you could not enforce a policy that all databases have to be in the Full recovery model because the *ALTER DATABASE* command executes outside the context of a transaction.

Policy Categories

Policy categories can be used to group one or more policies into a single compliance unit. If not specified, all policies belong to the *DEFAULT* category. To check or enforce policies, you create a subscription to one or more policies.

Subscription occurs at two levels—instance and database. A member of the sysadmin role can subscribe an instance to a policy category. Once subscribed, the owner of each database within the instance can subscribe their database to a policy category.

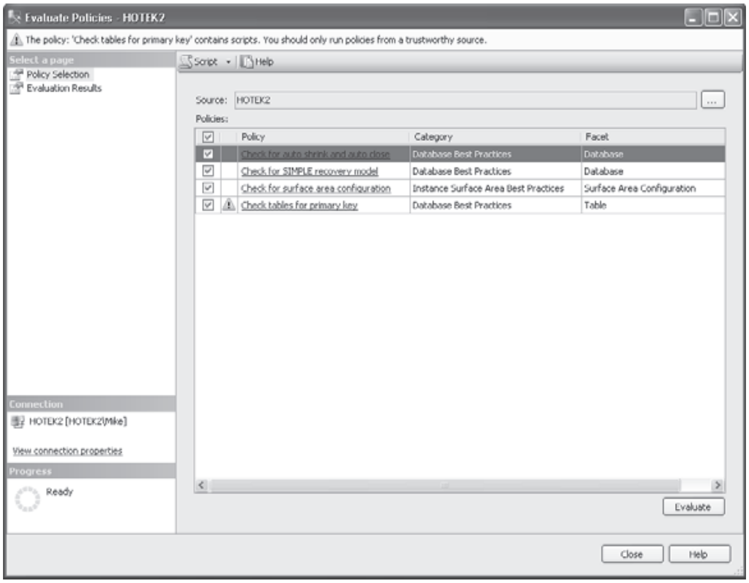
Each policy category has a *Mandate* property that applies to databases. When a policy category is set to *Mandate* and a sysadmin subscribes the instance to a policy category, all databases that meet the target set are controlled by the policies within the policy category. A policy subscription to a policy category set to *Mandate* cannot be overridden by a database owner.

Policy Compliance

Because you cannot set all policies to enforce compliance, you need to check policies manually that cannot be enforced on a regular basis. You view policies that apply to an instance by right-clicking the name of the instance within Object Explorer and selecting Policies, View.

You can check policies that apply to an instance by right-clicking the name of the instance within Object Explorer and selecting Policies, Evaluate.

You can check all policies within an instance, as shown in Figure 8-1, by right-clicking the Policies node and selecting Evaluate.



The policy: 'Check tables for primary key' contains scripts. You should only run policies from a trustworthy source.

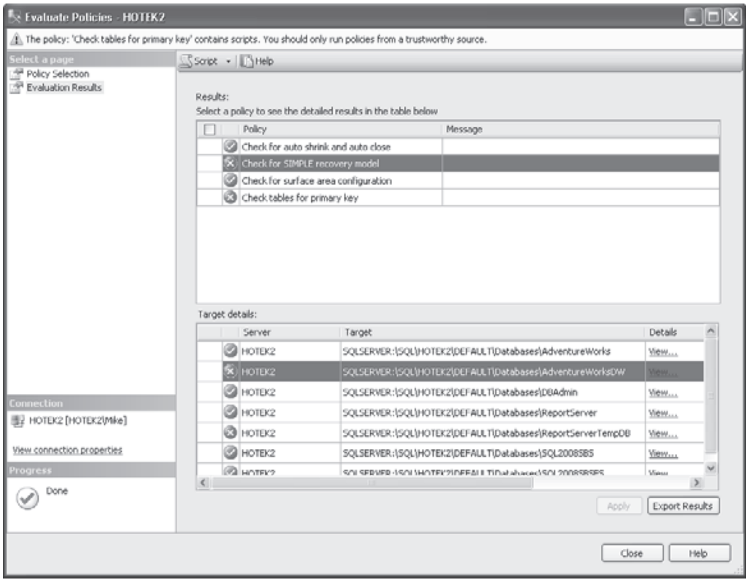
Source: HOTEK2

Policies:

☑		Policy	Category	Facet
☑		Check for auto shrink and auto close	Database Best Practices	Database
☑		Check for SIMPLE recovery model	Database Best Practices	Database
☑		Check for surface area configuration	Instance Surface Area Best Practices	Surface Area Configuration
☑	⚠	Check tables for primary key	Database Best Practices	Table

FIGURE 8-1 Evaluate policies

By clicking Evaluate, you execute the policies and review the results, as shown in Figure 8-2.



The policy: 'Check tables for primary key' contains scripts. You should only run policies from a trustworthy source.

Results:

Select a policy to see the detailed results in the table below

☐		Policy	Message
	✓	Check for auto shrink and auto close	
	✗	Check for SIMPLE recovery model	
	✓	Check for surface area configuration	
	✗	Check tables for primary key	

Target details:

		Server	Target	Details
	✓	HOTEK2	SQLSERVER:\SQL\HOTEK2\DEFAULT\Databases\AdventureWorks	View...
	✗	HOTEK2	SQLSERVER:\SQL\HOTEK2\DEFAULT\Databases\AdventureWorksDW	View...
	✓	HOTEK2	SQLSERVER:\SQL\HOTEK2\DEFAULT\Databases\DBAdmin	View...
	✓	HOTEK2	SQLSERVER:\SQL\HOTEK2\DEFAULT\Databases\ReportServer	View...
	✗	HOTEK2	SQLSERVER:\SQL\HOTEK2\DEFAULT\Databases\ReportServerTempDB	View...
	✓	HOTEK2	SQLSERVER:\SQL\HOTEK2\DEFAULT\Databases\SQL2008SBS	View...

FIGURE 8-2 Policy check results

EXAM TIP

Defining a condition to be used as a policy target is a critical component to well-defined policies. A policy fails during a check if the object does not conform to the criteria and if the property does not exist. For example, attempting to check that the Web Assistant is disabled against a SQL Server 2008 instance fails because the feature does not exist.

Central Management Server

Policy Based Management would be limited to SQL Server 2008 and be very tedious if you had to do any of the following:

- Duplicate policies on every instance
- Create subscriptions to each instance in your environment individually
- Check compliance for each instance individually

Within the Registered Servers pane in SSMS, you can configure a Central Management Server. Underneath the Central Management Server, you can create multiple levels of folders, and register instances into the appropriate folder. After you have the Central Management Server structure set up in SSMS, you can evaluate polices against a specific instance, folder, or all instances underneath the Central Management Server. Figure 8-3 shows an example of a Central Management Server.

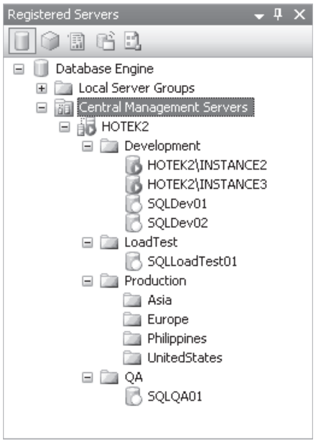


FIGURE 8-3 Central Management Server

Import and Export Policies

Policies and conditions can be exported to files as well as imported from files. SQL Server ships with 53 policies that are located in the Microsoft SQL Server\100\Tools\Policies folder. There are 50 policies for the database engine, 2 policies for Reporting Services, and 1 policy for Analysis Services. The CodePlex site (*http://www.codeplex.com*) has additional policies that you can download and import.

You can import policies within the Registered Servers pane or the Object Explorer. Within Object Explorer, you can right-click the Policies node underneath Policy Management and select Import Policy. Within Registered Servers, you can right-click the Central Management Server or any folder or instance underneath the Central Management Server and select Import Policies. If you import policies from the Central Management Server, the policies are imported to every instance defined underneath the Central Management Server, but not to the Central Management Server itself. Likewise, right-clicking a folder imports the policies to all instances within the folder hierarchy. To import policies to the Central Management Server, you must connect to the instance within Object Explorer and import from the Policies node.

✓ Quick Check

1. **What are the five objects that are used within Policy Based Management?**
2. **What are the allowed execution modes for a policy?**
3. **Which object has a property that allows you to mandate checking for all databases on an instance?**
4. **How many facets can be checked within a single condition?**
5. **How many conditions can be checked within a single policy?**

Quick Check Answers

1. **The objects that are used with Policy Based Management are facets, conditions, policies, policy targets, and policy categories.**
2. **The policy execution modes are *On demand*, *On schedule, On change, Log only*, and *On change, prevent*.**
3. **Policy categories allow you to mandate checking of all databases within an instance.**
4. **A condition can be defined on only one facet.**
5. **A policy can check only a single condition.**

Defining Policies and Checking for Compliance

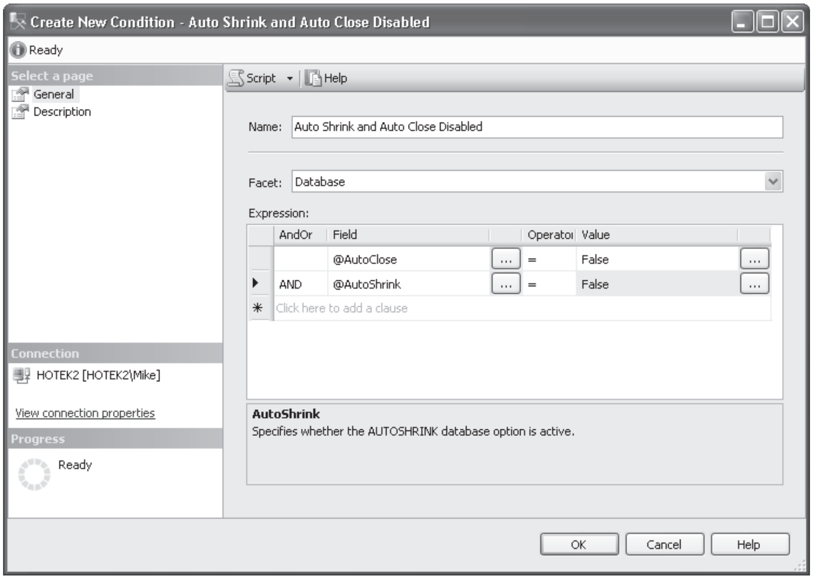
In these practices, you define and check several policies for your environment.

PRACTICE 1 Create a Condition

In this practice, you create a condition for the following:

- Check that a database does not have the *auto shrink* or *auto close* properties set.
- Check that CLR, OLE Automation, Ad Hoc Remote Queries, and SQL Mail are all disabled.
- Check that a database is not in the Simple recovery model.
- Check that all tables have a primary key.

1. In Object Explorer, expand the Policy Management node within the Management node.
2. Right-click the Conditions node and select New Condition.
3. Configure the condition as shown here. Click OK when you are done.



Name: Auto Shrink and Auto Close Disabled

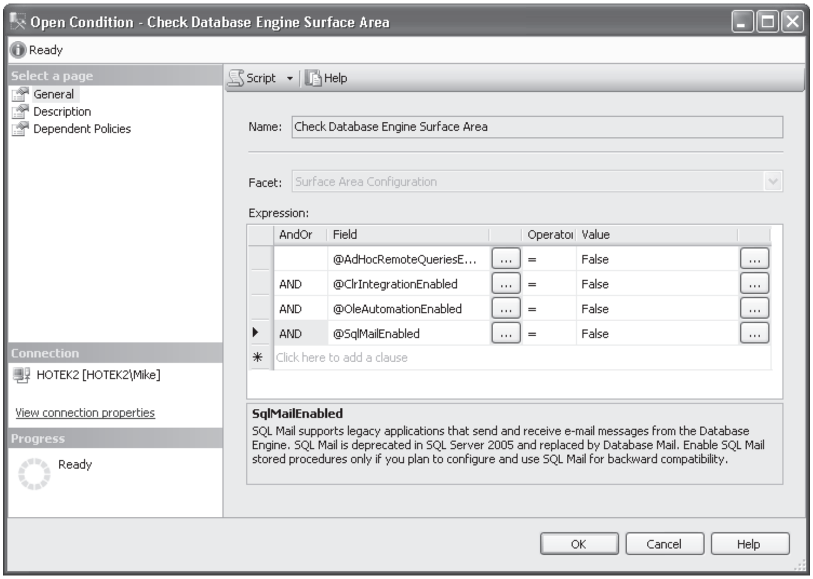
Facet: Database

Expression:

AndOr	Field	Operator	Value
	@AutoClose	=	False
AND	@AutoShrink	=	False
Click here to add a clause			

AutoShrink
Specifies whether the AUTOSHRINK database option is active.

4. Right-click the Conditions node again, select New Condition, and configure the condition as shown here. Click OK.



Name: Check Database Engine Surface Area

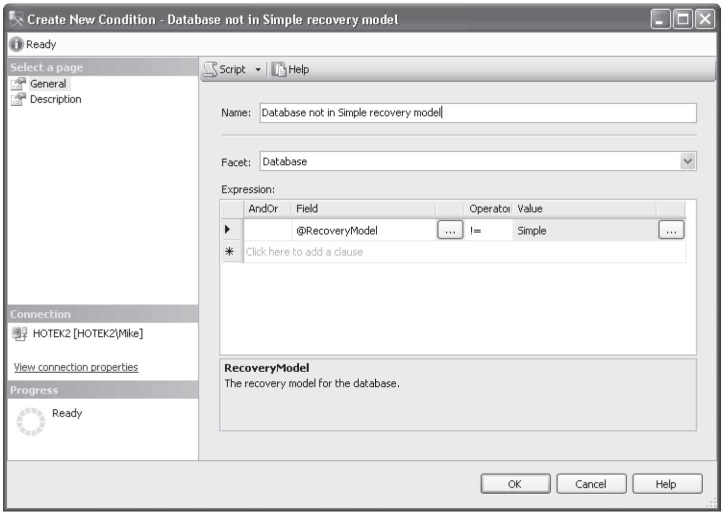
Facet: Surface Area Configuration

Expression:

AndOr	Field	Operator	Value
	@AdHocRemoteQueriesE…	=	False
AND	@ClrIntegrationEnabled	=	False
AND	@OleAutomationEnabled	=	False
AND	@SqlMailEnabled	=	False
Click here to add a clause			

SqlMailEnabled
SQL Mail supports legacy applications that send and receive e-mail messages from the Database Engine. SQL Mail is deprecated in SQL Server 2005 and replaced by Database Mail. Enable SQL Mail stored procedures only if you plan to configure and use SQL Mail for backward compatibility.

5. Right-click the Conditions node, select New Condition, and configure this third condition as shown here. Click OK when you are finished.



Name: Database not in Simple recovery model

Facet: Database

Expression:

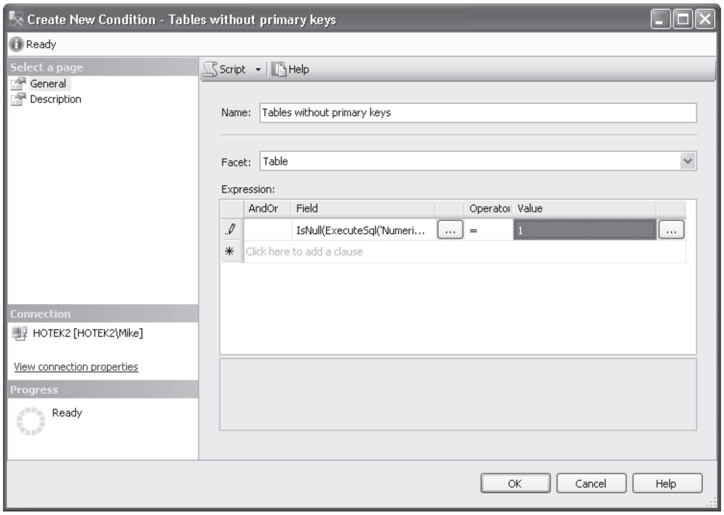
AndOr	Field	Operator	Value
	@RecoveryModel	!=	Simple
Click here to add a clause			

RecoveryModel
The recovery model for the database.

6. Right-click the Conditions node and select New Condition. Select the Table facet, click the ellipsis button next to the Field column to display the Advanced Edit dialog box, enter the following code in the Cell Value text box, and click OK:

```
IsNull(ExecuteSql('Numeric', 'SELECT 1 FROM sys.tables a INNER JOIN sys.indexes b
   ON a.object_id = b.object_id WHERE b.is_primary_key = 1
    AND a.name = @@ObjectName AND a.schema_id = SCHEMA_ID(@@SchemaName)'), 0)
```

7. Configure the Name, Operator, and Value as shown here, and then click OK.



Name: Tables without primary keys

Facet: Table

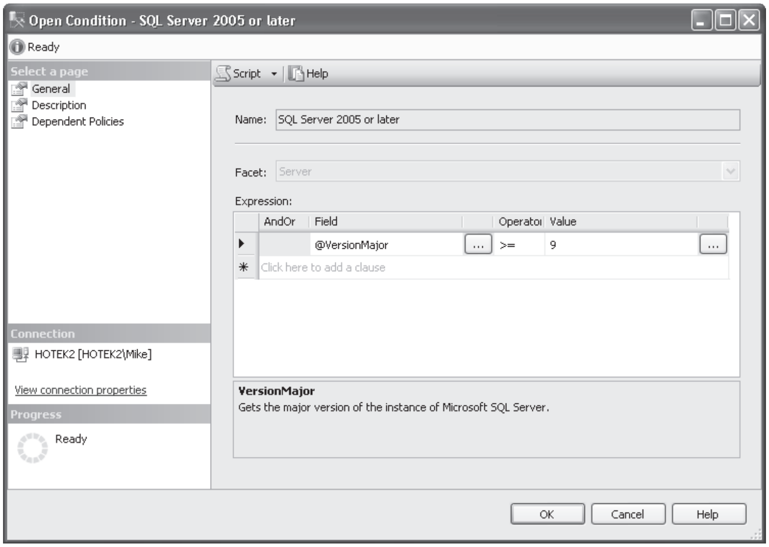
Expression:

AndOr	Field	Operator	Value
	IsNull(ExecuteSql('Numeri…	=	1
Click here to add a clause			

PRACTICE 2 Create a Condition for a Target Set

In this practice, you create a condition to target all SQL Server 2005 and later instances, along with a condition to target all user databases that are online.

1. Right-click the Conditions node, select New Condition, and configure the condition as shown here. Click OK.



Name: SQL Server 2005 or later

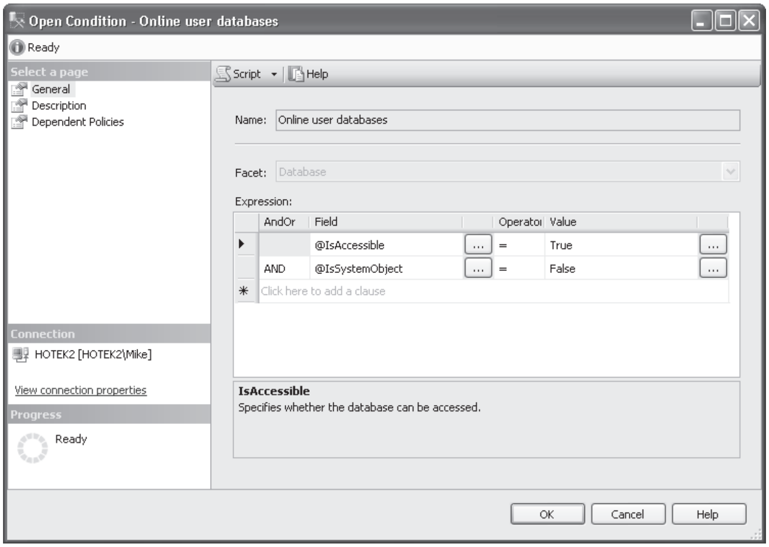
Facet: Server

Expression:

AndOr	Field	Operator	Value
	@VersionMajor	>=	9
Click here to add a clause			

VersionMajor
Gets the major version of the instance of Microsoft SQL Server.

2. Right-click the Conditions node, select New Condition, and configure the condition as shown here. Click OK when you are done.



Name: Online user databases

Facet: Database

Expression:

AndOr	Field	Operator	Value
	@IsAccessible	=	True
AND	@IsSystemObject	=	False
Click here to add a clause			

IsAccessible
Specifies whether the database can be accessed.

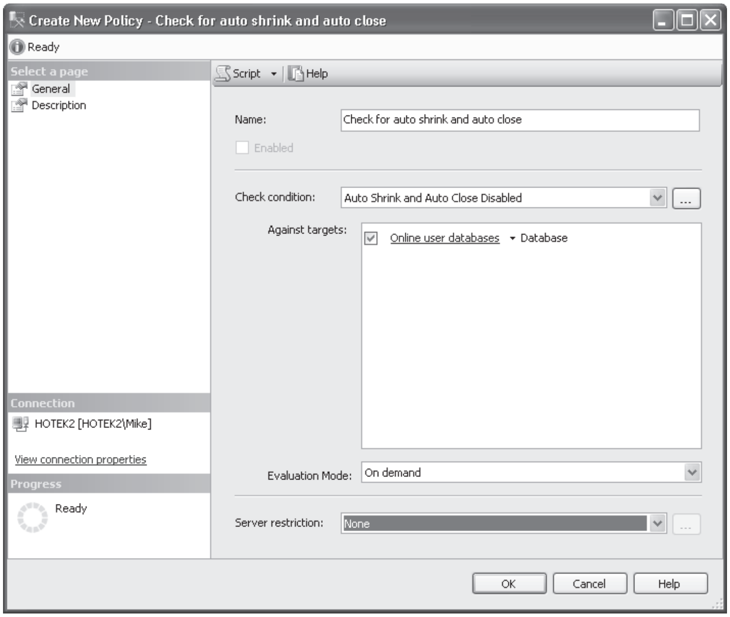
PRACTICE 3 Create a Policy

In this practice, you create policies that use the conditions you just created to do the following:

- Check that a database does not have the *auto shrink* or *auto close* properties set.
- Check that CLR, OLE Automation, Ad Hoc Remote Queries, and SQL Mail are all disabled.

- Check that a database is not in the Simple recovery model.
- Check that all tables have a primary key.

1. Right-click the Policies node, select New Policy, and configure the policy as shown here. Click OK.



Create New Policy - Check for auto shrink and auto close

Name: Check for auto shrink and auto close

☐ Enabled

Check condition: Auto Shrink and Auto Close Disabled

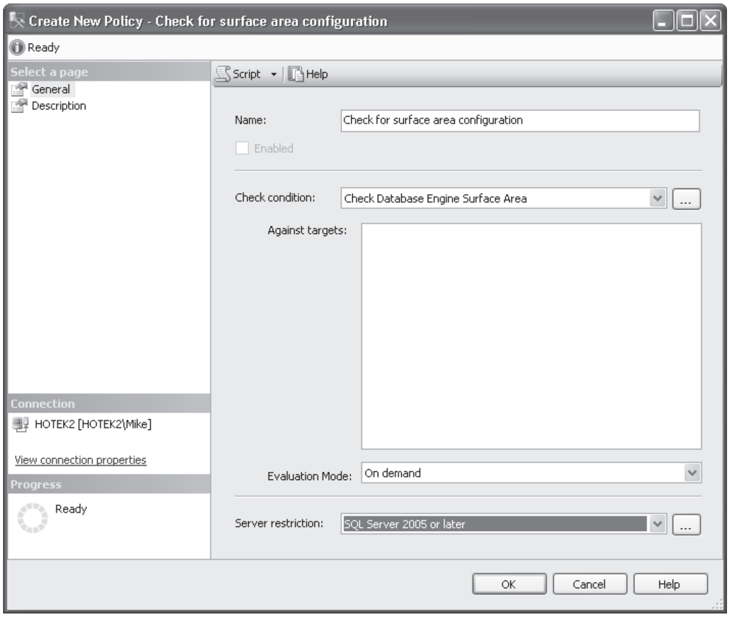
Against targets: ☑ Online user databases Database

Evaluation Mode: On demand

Server restriction: None

HOTEK2 [HOTEK2\Mike]

2. Right-click the Policies node, select New Policy, and configure this second policy as shown here. Click OK.



Create New Policy - Check for surface area configuration

Name: Check for surface area configuration

☐ Enabled

Check condition: Check Database Engine Surface Area

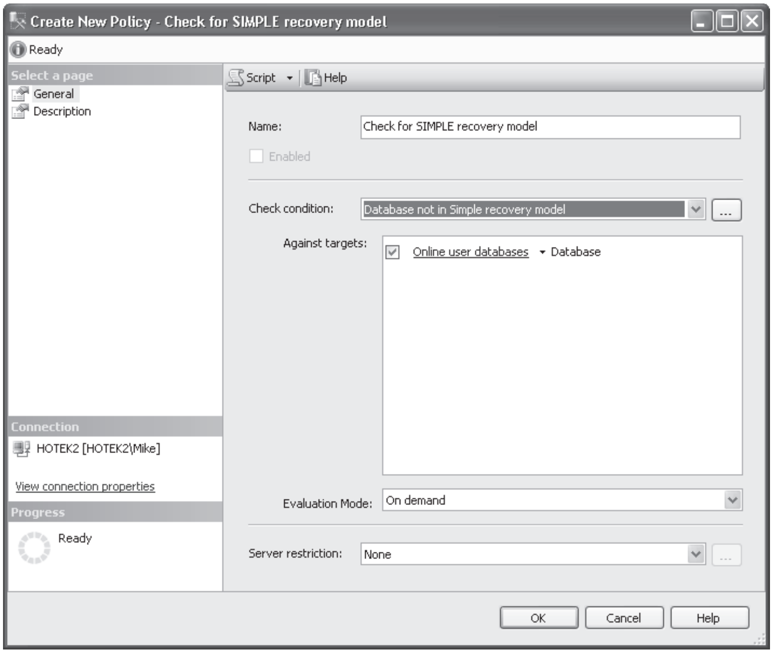
Against targets:

Evaluation Mode: On demand

Server restriction: SQL Server 2005 or later

HOTEK2 [HOTEK2\Mike]

3. Right-click the Policies node, select New Policy, and configure the policy as shown here. Click OK.



Create New Policy - Check for SIMPLE recovery model

Name: Check for SIMPLE recovery model

☐ Enabled

Check condition: Database not in Simple recovery model

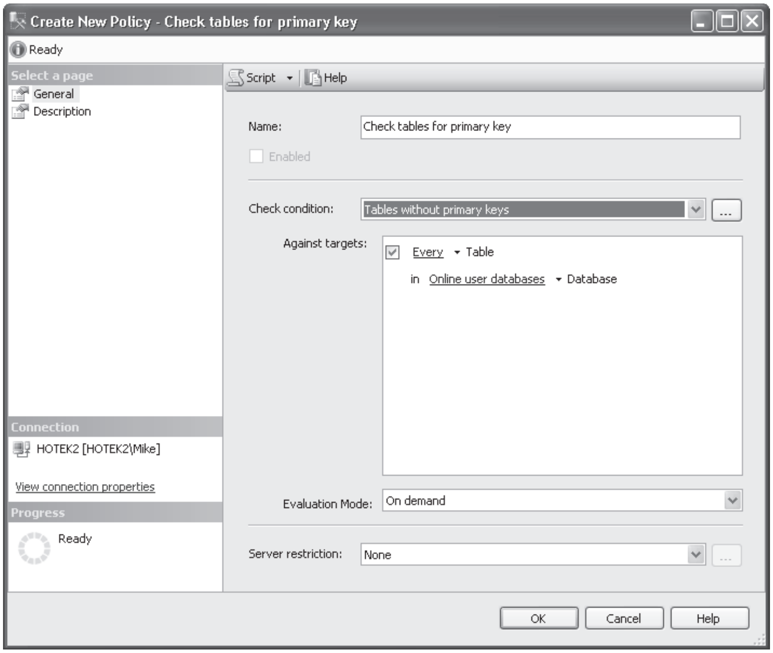
Against targets: ☑ Online user databases Database

Evaluation Mode: On demand

Server restriction: None

HOTEK2 [HOTEK2\Mike]

4. Right-click the Policies node, select New Policy, and configure the last policy as shown here. Click OK.



Create New Policy - Check tables for primary key

Name: Check tables for primary key

☐ Enabled

Check condition: Tables without primary keys

Against targets: ☑ Every Table
in Online user databases Database

Evaluation Mode: On demand

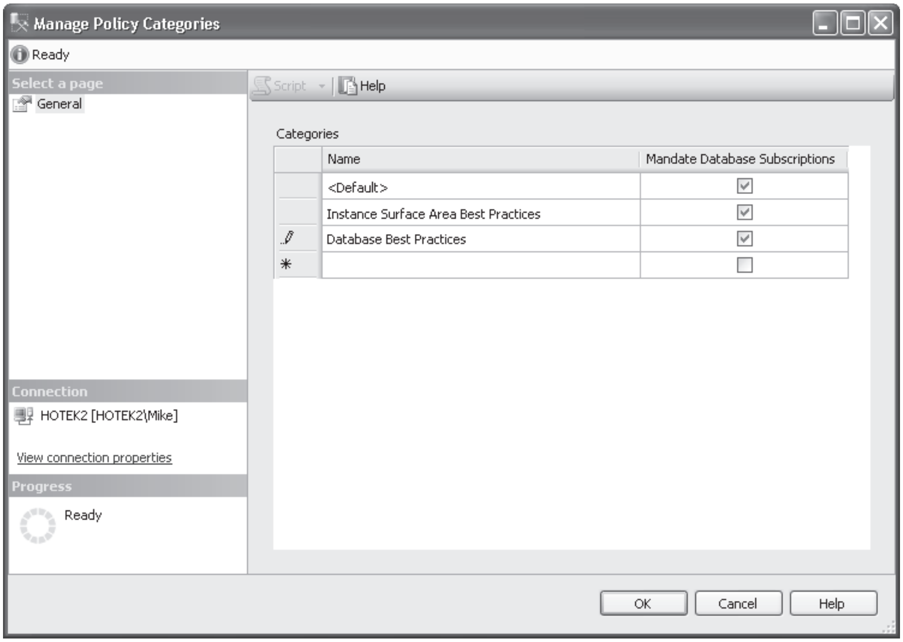
Server restriction: None

HOTEK2 [HOTEK2\Mike]

PRACTICE 4 Create a Policy Category

In this practice, you create two policy categories for the policies that you created.

1. Right-click Policy Management, select Manage Categories, and create the categories as shown here. Click OK.



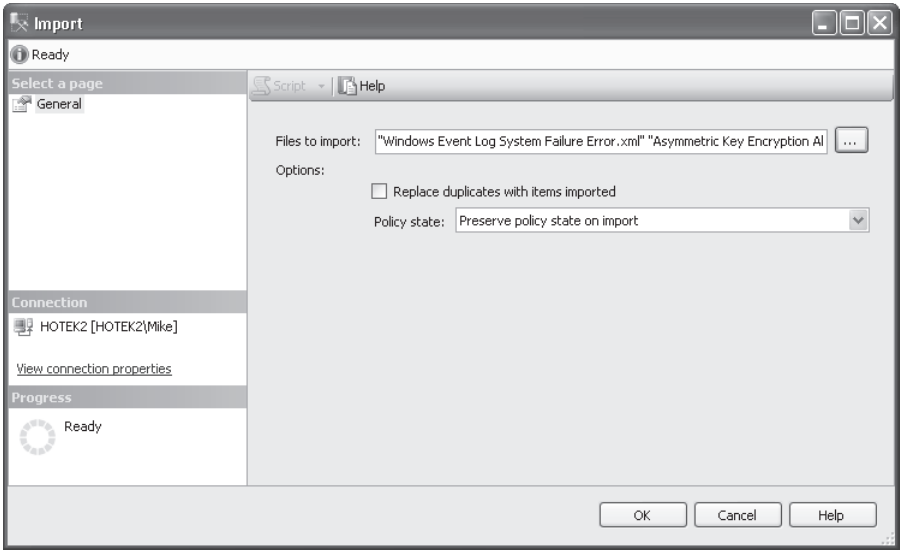
Name	Mandate Database Subscriptions
<Default>	☑
Instance Surface Area Best Practices	☑
Database Best Practices	☑
	☐

2. In SSMS, in the console tree, expand the Policies folder. Right-click the Check For Auto Shrink And Auto Close Policy, select Properties, click the Description tab, and change the category to Database Best Practices. Click OK.
3. Right-click the Check For Simple Recovery Model Policy, select Properties, select the Description tab, and change the category to Database Best Practices. Click OK.
4. Right-click the Check For Surface Area Configuration Policy, select Properties, click the Description tab, and change the category to Instance Surface Area Best Practices. Click OK.
5. Right-click the Check Tables For Primary Key Policy, select Properties, select the Description tab, and change the category to Database Best Practices. Click OK.

PRACTICE 5 Import Policies

In this practice, you import the policies that ship with SQL Server.

1. Right-click the Policies node underneath Policy Management and select Import Policy.
2. Click the ellipsis button next to the Files To Import text box, navigate to the Microsoft SQL Server\100\Tools\Policies\DatabaseEngine\1033 folder, select all the files in the folder, as shown here, and click Open.



Files to import: "Windows Event Log System Failure Error.xml" "Asymmetric Key Encryption Al

Options:

☐ Replace duplicates with items imported

Policy state: Preserve policy state on import

HOTEK2 [HOTEK2\Mike]

3. Select the Replace Duplicates With Items Imported check box, select Preserve Policy State On Import, and click OK.
4. Take the time to browse the policies and conditions that were created during the import.

Lesson Summary

- You can build policies to enforce conditions across any version of SQL Server.
- Policies can enforce a single condition and each condition can be based on a single facet.
- Policy categories allow you to group policies together for compliance checking.
- A policy category can be set with the *Mandate* property, which requires the policy to be checked against all databases within an instance.

Lesson Review

The following question is intended to reinforce key information presented in this lesson. The question is also available on the companion CD if you prefer to review it in electronic form.

NOTE **ANSWERS**

Answers to this question and an explanation of why each answer choice is correct or incorrect is located in the "Answers" section at the end of the book.

1. You have defined several policies that you want applied to all databases within an instance. How do you ensure that a database owner is not allowed to avoid the policy check with the least amount of administrative effort?
   - **A.** Create a condition that checks all databases.
   - **B.** Add the policy to a user-defined policy category and set the *Mandate* property.
   - **C.** Add the policy to the default policy category.
   - **D.** Check the policies manually against the instance.

Chapter Review

To practice and reinforce the skills you learned in this chapter further, you can perform the following tasks:

- Review the chapter summary.
- Review the list of key terms introduced in this chapter.
- Complete the case scenario. The scenario sets up a real-world situation involving the topics in this chapter and asks you to create a solution.
- Complete the suggested practices.
- Take a practice test.

Chapter Summary

- Facets are the .NET assemblies that define the set of properties for an object upon which conditions are built.
- A condition can be defined for a single facet and a policy can be checked for a single instance.
- Policies can be checked manually or automatically. Automatic policy checking can be performed on a scheduled basis or by using the event notification infrastructure.
- A database owner can subscriber a database to one or more policies; however, a policy that belongs to a policy category set with the *Mandate* property requires checking against all databases.

Key Terms

Do you know what these key terms mean? You can check your answers by looking up the terms in the glossary at the end of the book.

- Condition
- Facet
- Policy category
- Policy target

Case Scenario

In the following case scenario, you apply what you've learned in this chapter. You can find answers to these questions in the "Answers" section at the end of this book.

Case Scenario: Designing a Management Strategy for Coho Vineyard

BACKGROUND

Company Overview

Coho Vineyard was founded in 1947 as a local, family-run winery. Due to the award-winning wines it has produced over the last several decades, Coho Vineyards has experienced significant growth. To continue expanding, several existing wineries were acquired over the years. Today, the company owns 16 wineries; 9 wineries are in Washington, Oregon, and California, and the remaining 7 wineries are located in Wisconsin and Michigan. The wineries employ 532 people, 162 of whom work in the central office that houses servers critical to the business. The company has 122 salespeople who travel around the world and need access to up-to-date inventory availability.

Planned Changes

Until now, each of the 16 wineries owned by Coho Vineyard has run a separate Web site locally on the premises. Coho Vineyard wants to consolidate the Web presence of these wineries so that Web visitors can purchase products from all 16 wineries from a single online store. All data associated with this Web site will be stored in databases in the central office.

When the data is consolidated at the central office, merge replication will be used to deliver data to the salespeople as well as to allow them to enter orders. To meet the needs of the salespeople until the consolidation project is completed, inventory data at each winery is sent to the central office at the end of each day.

Management wants to ensure that you cannot execute stored procedures written in C#.NET or use the *OPENROWSET* or *OPENDATASOURCE* command.

EXISTING DATA ENVIRONMENT

Databases

Each winery presently maintains its own database to store all business information. At the end of each month, this information is brought to the central office and transferred into the databases shown in Table 8-1.

TABLE 8-1 Coho Vineyard Databases

DATABASE	SIZE
Customer	180 megabytes (MB)
Accounting	500 MB
HR	100 MB
Inventory	250 MB
Promotions	80 MB

After the database consolidation project is complete, a new database named *Order* will serve as a data store to the new Web store. As part of their daily work, employees also will connect periodically to the *Order* database using a new in-house Web application.

The *HR* database contains sensitive data and is protected using Transparent Data Encryption (TDE). In addition, data in the Salary table is encrypted using a certificate.

Database Servers

A single server named DB1 contains all the databases at the central office. DB1 is running SQL Server 2008 Enterprise on Windows Server 2003 Enterprise.

Business Requirements

You need to design an archiving solution for the *Customer* and *Order* databases. Your archival strategy should allow the *Customer* data to be saved for six years.

To prepare the *Order* database for archiving procedures, you create a partitioned table named Order.Sales. Order.Sales includes two partitions. Partition 1 includes sales activity for the current month. Partition 2 is used to store sales activity for the previous month. Orders placed before the previous month will be moved to another partitioned table named Order.Archive. Partition 1 of Order.Archive includes all archived data. Partition 2 remains empty.

A process needs to be created to load the inventory data from each of the 16 wineries by 4 A.M. daily.

Four large customers submit orders using Coho Vineyards Extensible Markup Language (XML) schema for Electronic Data Interchange (EDI) transactions. The EDI files arrive by 5 P.M. and need to be parsed and loaded into the *Customer, Accounting,* and *Inventory* databases, which each contain tables relevant to placing an order. The EDI import routine is currently a single threaded C++ application that takes between three and six hours to process the files. You need to finish the EDI process by 5:30 P.M. to meet your Service Level Agreement (SLA) with the customers. After the consolidation project finishes, the EDI routine loads all data into the new *Order* database.

You need to back up all databases at all locations. All production databases are required to be configured with the Full recovery model. You can lose a maximum of five minutes of data under a worst-case scenario. The *Customer, Account, Inventory, Promotions,* and *Order* databases can be off-line for a maximum of 20 minutes in the event of a disaster. Data older than six months in the *Customer* and *Order* databases can be off-line for up to 12 hours in the event of a disaster.

Answer the following question.

- What policies would you implement to check and enforce the business requirements for Coho Vineyard?

Suggested Practices

To help you master the exam objectives presented in this chapter, complete the following tasks.

Implement Policy Based Management

- **Practice 1** Configure a policy to check the surface area configuration for all your SQL Server instances.
- **Practice 2** Configure a policy to check the last time a database was successfully backed up.
- **Practice 3** Configure policies to check the membership of the sysadmin and db_owner roles.
- **Practice 4** Configure a policy to ensure that databases are not set to either *auto shrink* or *auto close*.
- **Practice 5** Based on the policies that ship with SQL Server 2008, decide which policies apply to your environment and implement the policy checks.

Take a Practice Test

The practice tests on this book's companion CD offer many options. For example, you can test yourself on just one exam objective, or you can test yourself on all the 70-432 certification exam content. You can set up the test so that it closely simulates the experience of taking a certification exam, or you can set it up in study mode so that you can look at the correct answers and explanations after you answer each question.

MORE INFO **PRACTICE TESTS**

For details about all the practice test options available, see the section entitled "How to Use the Practice Tests," in the Introduction to this book.